برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۱

مدرس: رامتین خسروی

طراحان: الهه خداوردی، شهریار عطار

مهلت تحویل: دوشنبه ۶ اسفند ۱۴۰۳، ساعت ۲۳:۵۹

مقدمه

هدف از این تمرین آشنایی شما با container-ها و iterator-هاست. در این تمرین شما یک نمونه ساده شده از یک برنامه مدیریت پروژه را پیاده‌سازی کنید.

علاوه بر اجرای درست برنامه، جدا کردن مسئولیت‌ها به تابع‌های مختلف، تمیزی کد، استفاده از container-ها و iterator-ها اهمیت زیادی دارند. سعی کنید در پیاده‌سازی تمرین، توابع مختلفی تعریف کنید که هر کدام تنها یک کار انجام می‌دهند.

سیستم مدیریت پروژه

Darth Vader پس از اینکه با موفقیت یک راه ارتباطی امن با کمک رمزگذاری ایجاد کرد شروع به دوباره‌سازی Death Star کرد، از آنجایی که این پروژه، بسیار بزرگ بود و مدیریت کردن آن سخت بود او نیازمند برنامه‌ای بود که به او در مدیریت این پروژه کمک کند. از ویژگی‌های کلی این برنامه می‌توان به تعریف کارهای جدید، اضافه کردن کارمندان، اختصاص دادن کارها و گزارش‌گیری اشاره کرد. توجه داشته باشید که تضمین می‌شود در همه بخش‌ها فرمت ورودی درست است و نیازی به بررسی خطاهای مربوط به این نوع نیست. همچنین تضمین می‌شود که نام کارها در زمان اختصاص دادن به کارمندان و انجام دادن، پیش از این تعریف شده است، و همین شرایط برای نام کارمندان نیز برقرار است، همچنین نام کارها و کارمندان **یکتا** است و **همگی با حروف کوچک** هستند.

تعریف کارها

با کمک این دستور ما یک کار جدید تعریف می‌کنیم. توجه داشته باشید در همه دستورات، هیچ یک از مقادیر حاوی whitespace نیست و کلمات چند بخشی با _[1] از هم جدا شده‌اند. در این دستور هر چه مقدار priority کمتر باشد، آن کار اولویت بالاتری دارد و باید زودتر انجام شود. وقتی کار جدید تعریف می‌شود در حالت TODO قرار می‌گیرد.

پیکربندی ورودی
`add_task <task_name> <priority>`

مثال ورودی
`add_task improve_db_performance 1`

اضافه کردن کارمندان

با کمک این دستور می‌توانیم یک کارمند جدید به سیستم خود اضافه کنیم.

پیکربندی ورودی
`add_employee <employee_name>`

مثال ورودی
`add_employee misagh`

[1] underscore

اختصاص دادن کارها

با کمک این دستور یک کار را به یکی از کارمندان محول می‌کنیم. پس از اختصاص دادن کار به کارمند وضعیت آن کار از TODO به ONGOING تغییر می‌کند. دقت کنید که ممکن است یک کار به بیش از یک کارمند محول شود، اما در هر بار وارد کردن این دستور فقط به یک کارمند می‌توانیم محول کنیم (به عبارت دیگر در صورتی که میخواهیم یک کار را به چند کارمند محول کنیم باید چند بار این دستور را وارد کنیم).

پیکربندی ورودی
`assign_employee <task_name> <employee_name>`

مثال ورودی
`assign_employee improve_db_performance misagh`

انجام دادن کارها

با کمک این دستور یک کار را به پایان می‌رسانیم. پس از انجام دادن کار، وضعیت آن کار از ONGOING به DONE تغییر می‌کند.

پیکربندی ورودی
`finish_task <task_name>`

مثال ورودی
`finish_task improve_db_performance`

نمایش گزارش

برای مطلع شدن از شرایط پروژه و نحوه پیشرفت، تعداد مختلفی گزارش داریم که به ازای هر کدام باید خروجی‌های مناسب را به کاربر نشان دهیم.

گزارش کلی

با کمک این دستور، تعداد کل کارها در هر نوع را می‌شماریم.

پیکربندی ورودی
`report all`

پیکربندی خروجی

`TODO: <number_of_todo_tasks>`
`ONGOING: <number_of_ongoing_tasks>`
`DONE: <number_of_done_tasks>`

مثال ورودی
`report all`

مثال خروجی

`TODO: 10`
`ONGOING: 3`
`DONE: 84`

گزارش کارهای در حال انجام

با کمک این دستور لیست تمامی کارهایی که در حال حاضر در حال انجام است را مشاهده می‌کنیم. خروجی‌ها باید به ترتیب اولویت (اولویت بالاتر ابتدا) مرتب شوند، و برای دو کاری که اولویت یکسان دارند بر اساس نام (کاری که از لحاظ الفبایی[2] زودتر بیاید) مرتب شوند. در خروجی ما باید ابتدا نام کار و سپس با یک فاصله اولویت آن کار را در پرانتز مشاهده کنیم، پس از آن باید لیست کارمندانی که در حال فعالیت بر روی این کار هستند به ترتیب الفبایی نمایش داده شوند.

پیکربندی ورودی
`report ongoing`

پیکربندی خروجی
`<task_name> (<priority>): name1, name2, name3` `<task_name_2> (<priority_2>): name1_2, name2_2`

مثال ورودی
`report ongoing`

مثال خروجی

`analyze_db_performance (1): low_mist, the_person`
`analyze_transactions (1): the_faint_faith`
`amortized_analysis (2): agha_amin, agha_aryan, agha_nemati`
`call_product_manager (5): misagh`

[2] alphabetical order

گزارش وضعیت یک کارمند

با کمک این دستور ما می‌توانیم عملکرد یک کارمند را بررسی کنیم. بعد از وارد کردن این دستور ما ابتدا خلاصه‌ای از عملکرد کارمند را مشاهده می‌کنیم و سپس کارهای در حال انجامش را مشاهده می‌کنیم. ترتیب مشاهده کارهای در حال انجام بر اساس ترتیبی است که کارها به کارمند محول شده (کاری که زودتر محول شده زودتر مشاهده می‌شود). توجه داشته باشید که خروجی به صورت یک لیست ترتیبی است، یعنی باید کنار هر کار شماره‌ای نیز جهت خوانایی و نشان دادن ترتیب برای آن بزنید(1، 2، ...) که کمی خواناتر باشد. اگر کارمند در حال حاضر کار در حال حاضری نداشته باشد خروجی نیز به شکل مثال دوم خواهد بود.

پیکربندی ورودی

```
report employee <employee>
```

پیکربندی خروجی

```
<employee> has done <number_of_done_tasks> tasks.
<employee> is currently working on these tasks:
1. <tasks_name>
2. <tasks_name_2>
...
```

مثال ورودی اول

```
report employee misagh
```

مثال خروجی اول

```
misagh has done 12 tasks.
misagh is currently working on these tasks:
1. integrating_components
2. tuning_db_performance
3. reviewing_latest_animes
```

مثال خروجی دوم

```
misagh has done 12 tasks.
misagh is currently not working on any tasks.
```

نکات و نحوه تحویل

- تحویل این تمرین در سامانه گیت‌هاب انجام می‌شود. برای انجام تمرین لطفا از طریق این لینک وارد شوید، پس از آن باید شماره دانشجویی خود را انتخاب کنید (دقت کنید که با کمک این شماره دانشجویی به شما نمره خواهیم داد، لطفا در انتخاب درست شماره دانشجویی حتما دقت کنید، در صورتی که به مشکل خوردید با دستیاران در ارتباط باشید). پس از آن به صفحه‌ای منتقل می‌شوید که در آنجا می‌توانید تمرین جدید را قبول کنید، پس از قبول کردن تمرین یک مخزن در AP Assignments برای شما ساخته می‌شود و باید کدهای خود را در آنجا قرار دهید. برای این تمرین نیاز نیست کار به طور کامل با گیت را بلد باشید و صرفا کد نهایی خود را حتما در فایل src/main.cpp قرار دهید.
- پس از انجام تمرین و بارگذاری در گیت‌هاب، کد Hash آخرین کامیت را به همراه شماره دانشجویی خود در سامانه ای‌لرن آپلود کنید (در خط اول شماره دانشجویی، پس از آن از Enter استفاده کنید و به خط بعد بروید و پس از آن Hash آخرین کامیت). نمونه متن خواسته شده در سامانه ای‌لرن (بخش <last_commit_hash> و <sid> را جایگزین کنید):

```
<sid>
<last_commit_hash>
```

نمونه:

```
810100000
bad8fbcdcfa3b9feb371a31e0c370150aa870b18
```

- دقت کنید که عدم رعایت ساختار گفته شده در آپلود یا تغییر ساختار فایل‌ها در مخزن (می‌توانید به دلخواه خود فایل اضافه کنید و ... اما اسم و ساختار فایل‌هایی که در ابتدا به شما داده می‌شود نباید تغییر کند) باعث کسر 5 درصد از نمره شما خواهد شد.
- برنامه شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++20 ترجمه و در زمان معقول برای ورودی های آزمون اجرا شود.
- **بخش مهمی از ارزیابی برنامه شما درستی عملکرد آن است. بنابراین به اندازه کافی ورودی‌های آزمایشی طراحی کنید تا درستی خروجی در حالت‌های مختلف آزموده شود و دقت کنید که اگر برنامه شما با مثال‌های داده شده درست کار کند لزوما به معنای درستی در تمام سناریوها نیست.**
- درستی برنامه شما از طریق آزمون‌های خودکار سنجیده می‌شود. به این ترتیب، لازم است خروجی تولید شده از نظر بزرگی و کوچکی حروف، رعایت فاصله‌ها، عدم وجود خروجی‌های اضافه، ... دقیقا مانند نمونه‌های داده شده باشد. بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff فرمت خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق سیاست درس با آن برخورد خواهد شد.

نمرات

- تمیزی کد
  - رعایت کردن نام‌گذاری صحیح و منسجم بودن[3]
  - عدم وجود کد تکراری
  - رعایت دندانه‌گذاری[4]
  - استفاده **صحیح** از متغیرهای ثابت[5] به جای Magic Value-ها[6]
  - استفاده **صحیح** از container-ها و iterator-ها به جای روش‌های قدیمی
  - استفاده مناسب از ساختار داده‌های مناسب
  - توابع کوتاه که فقط یک کار را انجام می‌دهند
- درستی کد
  - آزمون‌های خودکار

دقت کنید که موارد ذکر شده لزوما کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.

---

[3] Consistency

[4] Indentation

[5] Constant

[6] به مقادیر خاصی که در کد استفاده می‌شود و برای عملکرد صحیح کد ضروری است اما دلیل استفاده از آن‌ها مشخص نیست و قابل جایگزین شدن با یک ثابت با اسم مشخص جهت افزایش خوانایی هستند، magic value گفته می‌شود. برای آشنایی بیشتر با این مفهوم می‌توانید این لینک را مشاهده کنید.